Exhibit JX20

TON Development Status

Overall progress towards the test version: 70%

September 5, 2018

1. TON Virtual Machine (TVM)

TON VM or TVM is the component required for executing smart contracts in the TON Blockchain.

✓ *Implementation:* **95% complete**

TVM is fully implemented and internally tested. Minor modifications are likely to be necessary during the process of binding TVM with the TON Blockchain block generation and validation software.

In addition to TVM itself, a database required for storing on disk and accessing large amounts of TVM data (e.g., smart-contract code and data, old blocks, blockchain state) without loading all of it into memory has been developed.

✓ *Documentation:* **95% complete**

The current version of TVM is fully described in *Telegram Open Network Virtual Machine (September 5, 2018)*. Minor modifications to the implementation may require corresponding changes in the documentation.

2. TON Network

TON Network is the component required for delivering requests (e.g., proposed transactions) and propagating newly-generated TON Blockchain blocks through the network.

✓ *ADNL (low-level overlay network protocol running over IP networks):* **80% complete**

All functionality required for the test version is complete, including elliptic curve cryptography and the node lookup protocol. Some sophisticated options and additional cryptographic options that are not required for the launch of the test version will be implemented later prior to final launch.

✓ *Overlay networks over ADNL:* **100% complete**

Overlay networks are required to build node groups inside the ADNL networks. For instance, the validators for a shardchain create their separate overlay network to propagate new block candidates and run a Byzantine Fault Tolerant (BFT) consensus protocol.

✓ *Broadcast protocols for overlay networks over ADNL:* **100% complete**

Simple broadcast protocols are used inside overlay networks to propagate small messages, such as the BFT consensus protocol messages, to all members of an overlay network. These protocols are required for the implementation of validator BFT consensus.

✓ *CATCHAIN protocol:* **90% complete**

The CATCHAIN protocol is a variant of broadcast protocols tailored for implementing BFT consensus protocols and for solving similar group consensus tasks in a closed membership overlay network. As such, it is the first step in implementing the validator BFT consensus protocol.

✓ *Streaming broadcast protocols:* **95% complete**

Streaming broadcast protocols are used to quickly propagate large amounts of data, such as newly-generated TON Blockchain blocks (to all full nodes) and block candidates (to the validators of the corresponding shardchain). They use Forward Error Correction (FEC) protocols as their component.

3. TON Blockchain Block Generation and Validation

The block generation and validation software relies heavily on TVM and TON Network to create new block candidates, validate them between the validators, and propagate the signed blocks to all full nodes. Since the work on the TVM and TON Network components listed above is largely completed, the TON Blockchain is now in active development.

✓ *Documentation:* **90% complete**

The documentation is intended to present a complete description of the masterchain and shardchain block format in the TON Blockchain. Currently, we believe that the shardchain blocks are fully described in *Telegram Open Network Blockchain (September 5, 2018)*. Some details of the masterchain blocks, such as the list of all configurable parameters with their respective types, are not fully documented at this time because they are likely to be modified during the final development of the test version of the TON Blockchain.

▷ *Block manipulation library:* **30% complete**

The block manipulation library is intended to store entire blocks and their parts in files, load these data into memory, and access or modify different data structures present in a block. Most of the preparatory work has been accomplished, and this component is currently being implemented in our internal test version.

▷ *Validator software:* **10% complete**

Validator software uses the block manipulation library to generate block candidates, validate block candidates proposed by other validators, and achieve consensus on the next block in a shardchain or the masterchain. We have developed general, preliminary descriptions of this software. The validator software code will be written once the block manipulation library is completed.

▷ *Smart contract development, test, and debug environment:* **50% complete**

A test and debug environment for smart contracts is already implemented and internally tested, along with a low-level "TVM assembly" smart-contract language. The compiler from a high-level smart-contract language is 20% complete.

▷ *Fundamental and sample smart contracts:* **10% complete**

Some sample smart contracts are prepared in "TVM assembly". The implementation of fundamental smart contracts—which reside in the masterchain and run crucial tasks such as electing new validators and changing configurable parameters—requires the availability of the high-level smart-contract compilers and development tools discussed above. Active development of these smart contacts will begin immediately once the requisite compilers and development tools are in place.